A Training Scheme for Pattern Classification Using Multi-layer Feed-forward Neural Networks

Kanad Keeni*, Kenji Nakayama †, and Hiroshi Shimodaira ††
* Dept. of Information Systems and Quantitative Sciences
Nanzan University, Japan, keeni@iq.nanzan-u.ac.jp
† Dept. of Elec. & Comp. Eng., Kanazawa University. Japan
†† School of Information Science
Japan Advanced Institute of Science and Technology, Japan

Abstract

This study highlights on the subject of weight initialization in multi-layer feed-forward networks. Training data is analyzed and the notion of *critical point* is introduced for determining the initial weights for the input to hidden layer synaptic connections. The proposed method has been applied to artificial data. The experimental results show that the proposed method takes almost 1/2 of the training time required for standard back propagation.

1 Introduction

Neural networks architectures have sparked of great interest in recent years because of their intriguing learning capabilities. Several learning algorithms have been developed for training the networks and out of them Back Propagation [1] is probably most widely used. The reason for the popularity is the underlying simplicity and relative power of the algorithm. Its power derives from the fact that unlike its precursors, the perception learning rule [2], and the Widrow-Hoff learning rule [3], it can be employed for training nonlinear networks of arbitrary connectivity. Since such networks are often required for real-world applications, such a learning procedure is critical.

Now, in case of multi-layer neural networks (MLNN), the network is fed with the training data and during the learning process, it adjusts the synaptic weights and finds the optimal solution. However, the degree of freedom related to the selection of proper parameter is very high. They include the targets corresponding to the network outputs, initial weights, nonlinear functions of neurons, learning rates. Now, each of these factors play a crucial role in learning. In [4]-[6], it has been shown that the choice of targets largely effects generalization. Wilson has proposed Fast BPN [7], where the initial weights are determined by estimating the signal rank with generalized likelihood ratio test (GLRT) and the singular value decomposition (SVD) of the GLRT covariance matrix. However, the disadvantage of their method is the fact that the number of hidden nodes cannot exceed the input feature dimension. On the other hand, it has been shown in [8] that training data selection largely affects the generalization performance of networks.

This paper is divided into 5 sections. The next section describes the pattern mapping characteristics of feed-forward MLNN. The notion of *critical points* for generating initial weights from input to hidden layer synaptic connections is introduced in the third section. Experimental results of artificial data are provided in the fourth section. Finally, the last section is devoted to conclusion and further researches.

2 Pattern mapping characteristics of MLNN

If we assume that there are no overlaps among the distribution of training patterns belonging to different categories, then pattern mapping can be categorized in the following classes.

1. $\|\boldsymbol{X}_i - \boldsymbol{X}_j\|$ is small $\bigwedge$ $\|\boldsymbol{Y}_i - \boldsymbol{Y}_j\|$ is small
2. $\|\boldsymbol{X}_i - \boldsymbol{X}_j\|$ is small $\bigwedge$ $\|\boldsymbol{Y}_i - \boldsymbol{Y}_j\|$ is large
3. $\|\boldsymbol{X}_i - \boldsymbol{X}_j\|$ is large $\bigwedge$ $\|\boldsymbol{Y}_i - \boldsymbol{Y}_j\|$ is small
4. $\|\boldsymbol{X}_i - \boldsymbol{X}_j\|$ is large $\bigwedge$ $\|\boldsymbol{Y}_i - \boldsymbol{Y}_j\|$ is large

Here, $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ belong to class ω_1 and ω_2, $\boldsymbol{Y}_i$ and $\boldsymbol{Y}_j$ are the corresponding output vectors, and $\|\cdot\|$ stands for the Euclidean norm. In case of 1., the problem is to map similar input vectors in a way such that the corresponding output vectors also become similar. In the second case, the input vectors are similar but they are to be mapped as different patterns in the output space. The third case implies that the input patterns that are far from each other in the input space are to be mapped as similar patterns in the output space. Finally, the 4th

*A part of this research was sponsored by the Pache I A grant of Nanzan University, Nagoya, Japan

case means that the input patterns are far from each other in the input space and they are to be mapped as different patterns in the output space.

Now, the pattern mapping of 1., 3., and 4. are not that difficult. However, in case of 2., the problem is to map the patterns that are very close in the input space, as different patterns in the output space. In this case even though the solution exists, due to the difficulty of the problem the training process would be time consuming. Therefore, the second type of pattern mapping results in very slow learning and the possibility of arriving at a local solution is very high.

The proof for the above mention phenomenon is as follows.

If we define connection weight from the i'th input to j'th hidden unit as w_{ij} then the total input and the output of j'th hidden unit can be defined as follows.

$$net_j = \sum_{i=1}^{n} w_{ij}x_i + \theta_j, O_j = \sigma(net_j),$$

$$\sigma(net_j) = \frac{1}{1+e^{-net_j}}$$

where, $\sigma(\cdot)$ is the activation function and θ_j is the bias. At the same time the total input to the kth output unit and the corresponding output can be defined as follows.

$$net_k = \sum_{j=1}^{j} w_{jk}O_i + \theta_k, O_k = \sigma(net_k)$$

where, $\sigma(\cdot)$ is the same activation function as it was with the hidden layer.

Suppose we have training patterns $\boldsymbol{x}_{1n}$ and $\boldsymbol{x}_{2\acute{n}}$ that are very close in the input space and the patterns belong to the class ω_1 and ω_2 respectively. In this case, the network output would become extremely sensitive. This is because the network output must change rapidly for a small change in the input.

Now, if the decision boundary is far from the patterns $\boldsymbol{x}_{1n}$ and $\boldsymbol{x}_{2\acute{n}}$, then the corresponding outputs would have the value $O_{1n} \cong O_{2\acute{n}} \cong 0$ or 1. However, during the learning process, as the decision boundary approaches $\boldsymbol{x}_{1n}$ and $\boldsymbol{x}_{2\acute{n}}$ the output of the corresponding patterns also approach the same value, hence the learning process becomes extremely slow. In this case, as the decision boundary moves close to the pair $\boldsymbol{x}_{1n}$, $\boldsymbol{x}_{2\acute{n}}$ or enters the region between the pair, the amount of weight correction becomes extremely small. To be specific, if we assume $O_{1n} \cong O_{2\acute{n}} \cong$ some value y then the amount of correction for the n'th pattern Δ_n would be as follows.

$$\Delta_n = \eta\delta_n O_{nj}, \delta_n = (t_n - O_n)\acute{f}(net_n)$$

where, O_{nj} is the output of the j'th hidden unit. Now, as the patterns $\boldsymbol{x}_{1n}$ and $\boldsymbol{x}_{2\acute{n}}$ are similar, the output of the jth hidden unit would also become similar, that is

$$O_{1nj} \cong O_{2\acute{n}j}, \; and \; \acute{f}(net_{1n}) \cong \acute{f}(net_{2\acute{n}}).$$

In this case, the weight correction will be as follows.

$$\Delta_{1n}+\Delta_{2\acute{n}} = \eta O_{1nj}((t_{1n}-O_{1n})+(t_{2\acute{n}}-O_{2\acute{n}}))\acute{f}(net_{1n})$$

If it is assumed that the targets of the patterns are

$$t_{1n} = 1, t_{2\acute{n}} = 0$$

and the output of the patterns are

$$O_{1n} = z, O_{2\acute{n}} = z - \epsilon,$$

then the weight correction would become as follows.

$$\Delta_{1n}+\Delta_{2\acute{n}} = \eta O_{1nj}((t_{1n}-z)+(t_{2\acute{n}}-(z-\epsilon)))\acute{f}(net_{1n})$$

$$= ((t_{1n}+t_{2\acute{n}})-2z+\epsilon)\acute{f}(net_{1n}) = (1-2z+\epsilon)\acute{f}(net_{1n})$$

Now at beginning of training, the decision boundary would be far from $\boldsymbol{x}_{1n}$ and $\boldsymbol{x}_{2\acute{n}}$ and in that case the correction of synaptic weights would not be small. However, during the training process, as the decision boundary moves towards $\boldsymbol{x}_{1n}$ and $\boldsymbol{x}_{2\acute{n}}$, because of the similarity of the patterns the output would approach the same value. The most critical situation would take place as the value of z and $\|\epsilon\|$ approaches the value 0.5 and 0 respectively. That is,

$$\Delta_{1n} + \Delta_{2\acute{n}} = \lim_{z\to 0.5}\lim_{\epsilon\to 0}(1 - 2z + \epsilon)\acute{f}(net_{1n}) \cong 0$$

Therefore, the correction of weights for these patterns would become very small and as a result the learning process would become extremely slow.

On the other hand, if the patterns $\boldsymbol{x}_{1n}$ and $\boldsymbol{x}_{2\acute{n}}$ are far from each other in the input space, even if the decision boundary moves towards them the activation of the corresponding outputs would not become the same at the same time. Hence, the weight correction will not become small.

3 Optimization of initial weights

Here, the decision rule is to select the class corresponding to the output neuron with the largest output. For the sake of simplicity, the number of output unit is set to two (two-class classification

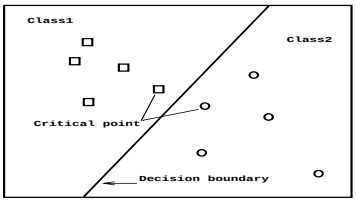

Figure 1: Critical points and decision boundary

problem). However, the concept can be hopefully extended to multi-class classification problems. The decision boundary for a multi-layer feed-forward network is defined as follows.

Definition 1. The decision boundary between two classes in a feed-forward neural networks is the locus of points where both of the output neurons produce the same activation.

If we define the activation output unit i as $O_i(\boldsymbol{x})$ where $\boldsymbol{x}$ is an input vector and let $d(\boldsymbol{x}) = O_1(\boldsymbol{x}) - O_2(x)$, then the decision boundary can be defined as

$$\{\boldsymbol{x} | d(\boldsymbol{x}) = 0\}$$

Next, the notion of *Critical points* is introduced as follows.

*Definition 2.*If we denote the samples in class ω_1 as $\boldsymbol{p}_i$ and samples in class ω_2 as $\boldsymbol{q}_j$ then for each sample in class ω_1 and class ω_2, the set of critical points $\boldsymbol{C}$ can be defined as

$$\boldsymbol{C} = \{(\boldsymbol{p}_i, \boldsymbol{q}_j) | \min_k(d(\boldsymbol{p}_i, \boldsymbol{q}_k)) = d(\boldsymbol{p}_i, \boldsymbol{q}_j),$$

$$\min_k(d(\boldsymbol{p}_k, \boldsymbol{q}_j)) = d(\boldsymbol{p}_i, \boldsymbol{q}_j), \boldsymbol{p}_i \in \omega_1, \boldsymbol{q}_j \in \omega_2\}$$

where $d(\boldsymbol{p}_i, \boldsymbol{q}_k)$ denotes the Euclidean distance between the vector $\boldsymbol{p}_i$ and $\boldsymbol{q}_k$. Now, as shown in Figure 1, the decision boundary must pass through the critical points. Now, as far as learn-ability is concerned, these critical points would play a very important role in learning. This is because the pair of critical points are patterns that are very close to each other in the input space and the mapping of these patterns correspond to the second type of pattern mapping discussed in the second section. Here the basic idea is to separate the similar patterns (in this case the pair of critical points) in the hidden layer from the very beginning of the learning. The details of the initial weight optimization procedure is as follows.

Since, the weight vectors are orthogonal to the separating hyper-plane, the initial weights are generated in the following way. First, the pair of critical points are determined from the training data as mentioned above. Next for all pair of critical points $(\boldsymbol{p}_i, \boldsymbol{q}_k)$ the weight vectors $\boldsymbol{m}_n$ are generated by the following equation:

$$\boldsymbol{m}_n = \frac{\boldsymbol{p}_i - \boldsymbol{q}_k}{\|\boldsymbol{p}_i - \boldsymbol{q}_i\|}$$

and the biases θ_n are generated by the following equation:

$$\theta_n = -\boldsymbol{n}^t \cdot \boldsymbol{P} = -\frac{(\boldsymbol{p}_i - \boldsymbol{q}_k)^t}{\|\boldsymbol{p}_i - \boldsymbol{q}_k\|} \cdot \frac{(\boldsymbol{p}_i + \boldsymbol{q}_k)}{2}$$

However, the proposed method would produce a large number of critical points. Hence, some kind of mechanism for the selection of critical points is necessary.

3.1 Selection criterion of critical points

As it has been mentioned previously, the critical points are the points that stay very close to each other and effect the whole learning process to a great extent. Therefore, the first criterion for selecting the pair of critical points based on the minimum distance among all the pairs is reasonable. However, this kind of approach is local, in the sense that a large number of critical points where the distance among each pair is very small, may appear very close to each other in the input space. Now, if the characteristic of the hyper-planes formed by the sigmoid function is considered, it is unrealistic to place a hyper-plane for each of these critical points. Therefore, some kind of global solution for selecting the critical point is indispensable.

Global Solution

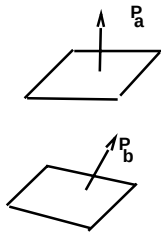

Figure 2: Correlation of hyper-planes

Suppose, we have two hyper planes $P_{\boldsymbol{a}}$ and $P_{\boldsymbol{b}}$ as shown in Figure 2. Now, if we express the equation of the hyper-planes as:

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n + a_0 = 0,$$
$$b_1 x_1 + b_2 x_2 + \cdots + b_n x_n + b_0 = 0$$

and the component vector $\boldsymbol{a}$ of $P_{\boldsymbol{a}}$ and $\boldsymbol{b}$ of $P_{\boldsymbol{b}}$ as

$$\boldsymbol{a} = (a_1, a_2, \ldots, a_n),\ \boldsymbol{b} = (b_1, b_2, \ldots, b_n)$$

respectively, where,

$$\textstyle\sum_{i=1}^n a_i^2 = 1, \text{ and } \sum_{i=1}^n b_i^2 = 1.$$

In this case, the first pair of $(\boldsymbol{p}_i, \boldsymbol{q}_k)$ is selected based on the minimum distance among all pairs of

critical points. In the next step, the previously selected critical points pair is ignored and the correlation of the previously selected pair and all the other remaining critical points are considered in the following way.

Suppose, $P_{\boldsymbol{a}}$ is the hyper plane calculated from the first pair of critical points $(\boldsymbol{p}_i, \boldsymbol{q}_k)$ and $Q_{\boldsymbol{b}}$ is the hyper-plane with which the correlation of the first hyper-plane is to be compared. So, in this case we will have two hyper-planes as shown in Figure 2.

In this case, the correlation of $P_{\boldsymbol{a}}$ and $Q_{\boldsymbol{b}}$ can be defined as:

$$D(P_{\boldsymbol{a}}, Q_{\boldsymbol{b}}) = \frac{\boldsymbol{a}^t \boldsymbol{b}}{\|\boldsymbol{a}\|\|\boldsymbol{b}\|}$$

However, the above mentioned correlation is still not sufficient for selecting the hyper-planes in the sense that there is a possibility of rejecting hyper-planes that are parallel to each other. Therefore, prior to selecting the next hyper-plane $Q_{\boldsymbol{b}}$, the distance $d_{midp}(P_{\boldsymbol{a}}, Q_{\boldsymbol{b}})$ is considered. Here, $d_{midp}(P_{\boldsymbol{a}}, Q_{\boldsymbol{b}})$ is the Euclidean distance between the locus of the center of the pair of critical points that represent the hyper-panes $P_{\boldsymbol{a}}$ and $Q_{\boldsymbol{b}}$. Finally, if $D(P_{\boldsymbol{a}}, Q_{\boldsymbol{b}})$ is $< max_\theta$ and $d_{midp}(P_{\boldsymbol{a}}, Q_{\boldsymbol{b}})$ is $< \delta$ σ then $Q_{\boldsymbol{b}}$ is merged with $P_{\boldsymbol{b}}$ otherwise $Q_{\boldsymbol{b}}$ is also selected and the process is repeated for all the other remaining critical points. Here, σ is standard deviation of the mid-points of the pair of critical points and δ is a parameter.

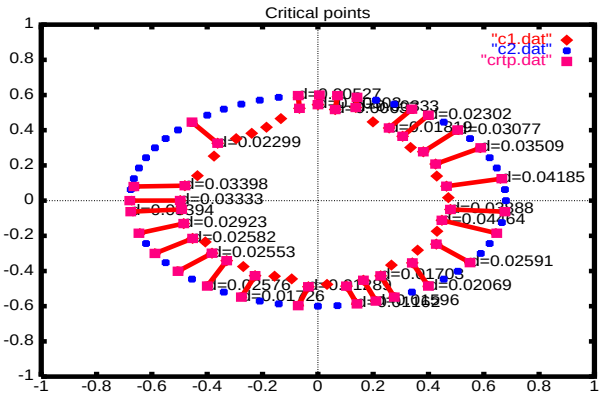

Figure 3: Calculated critical points

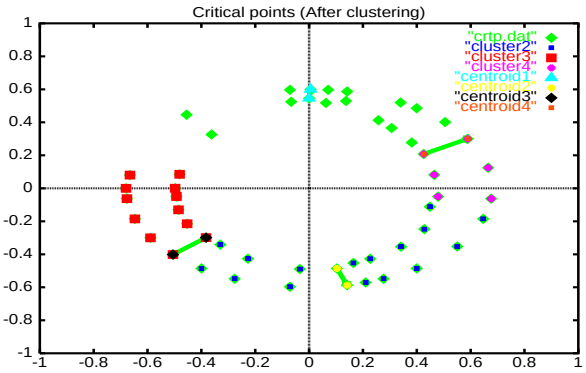

Figure 4: Critical points (After clustering)

4 Experiments

In order to demonstrate the effectiveness of the proposed critical points selection algorithm, the following problem has been set up. In this case, the training samples $\boldsymbol{x}_i$ of class ω_1 and $\boldsymbol{y}_i$ of class ω_2 lie on the circumference of a circle and an ellipse centered at (0,0) with radius $r1$ and the length of the major and minus axis of the ellipse is 0.7 and 0.6. Here, $r1$ satisfies the condition, $0.5 \leq r1 < 0.6$. . All of the critical points pairs are shown in Fig. 3. In reality only four hyper-planes are necessary for classifying the training patterns. The proposed method produced 26 critical points in the first phase and in the next phase it could give the minimum number of hidden units by merging the hyper-planes. The critical points after applying the proposed clustering method is shown in Fig. 4. As can be seen in Fig. 4, the proposed critical point selection method produced 4 hyper-planes and therefore the number of hidden units was set to 4. The calculated centroid vectors are connected by arrows.

Training was continued until the mean square error reach 0.001. For testing, 20000 samples were randomly generated and the class to which the testing sample falls is decided by considering the maximum activation of the output units. The decision boundary is estimated from output activation of the network in respect to the testing samples as mentioned in the previous subsection and it is shown in Fig. 5. On the other hand, another network was trained by employing standard back-propagation and the decision boundary is shown in Figure. 6.

Next, the network was trained with 10 different initial weights (weights for hidden to output unit connections) , and the average number of iterations taken by the proposed method and standard back-propagation are summarized in Table. 1.

Now, in order to investigate the effectiveness of the proposed critical points selection method, experiments were performed by employing the pairs based on minimum/maximum distance and the results are also shown in Table 1. It can be seen in Table 1 that the iterations necessary for the proposed method is almost half of the iterations required for standard back propagation. On the other hand, due to the peculiarity of the problem, if the critical points are selected based on minimum distance, then all the pairs would be very close to each other, and the results show that it takes more time compared to the proposed critical points selection

criterion. In case of standard back-propagation, there is no other way than to cut and try for determining the number of hidden units necessary for solving a problem. In case of the proposed method, the number of hidden unit can be determined automatically by controlling the parameters θ and δ.

Method	Average # Epoch
Critical points (Criterion 1)	712
Critical points (Criterion 2)	1213
Critical points (Criterion 3)	1277
Standard BP (Random weights)	1367

Table 1: Comparison of results

Criterion1 : max_θ and $d_{midp}(P_{\boldsymbol{a}}, Q_{\boldsymbol{b}})$
Criterion2 : First four pairs based on d_{max}
Criterion3 : First four pairs based on d_{min}

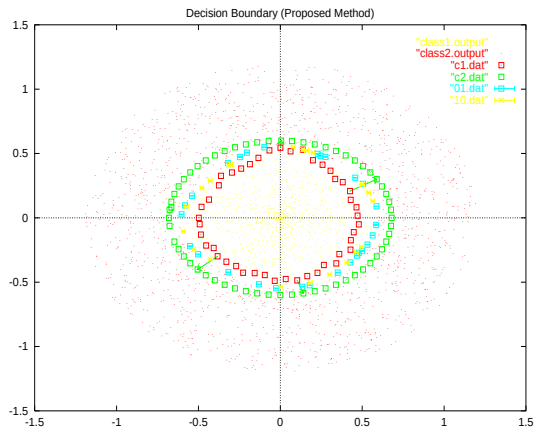

Figure 5: Decision boundary given by proposed method

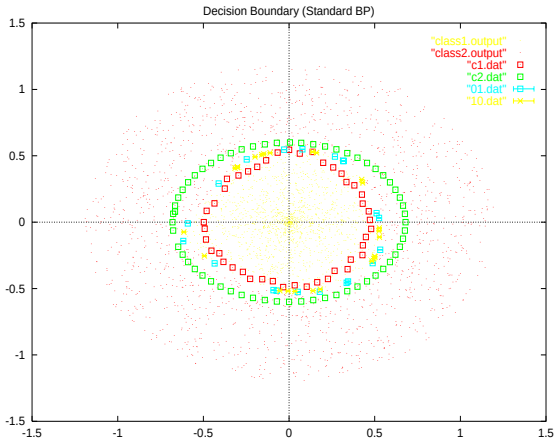

Figure 6: Decision boundary given by Conventional Back-propagation

5 Conclusion and further researches

It has been successfully shown through experiments that the a priori related to decision boundary can be employed for determining the initial weights of the network. The notion of critical points has been introduced for determining the initial weights. Compared to standard back-propagation the proposed method reduces training time. The method has to be further applied to real data for estimating its effect on generalization performance.

References

[1] Rumelhart, McClelland, and the PDP Research Group, "Parallel Distributed Processing," *The MIT Press*, 1989.

[2] Rosenblatt,F : Two Theorems of Statistical Separability in the Perceptron;*Proceedings of a Symposium on the Mechanization of Thought Process, Her Majesty's Stationary Office, London*,1959,421-456.

[3] Widrow, B., and Hoff, M.E: Adaptive Switching Circuits;Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record,part4,1960,96-104.

[4] K. Keeni, T. Nishino, H. Shimodaira, "On Feature Extraction of Indian Characters by Using Neural Networks," *IEICE*, Technical report of IEICE, COMP94-24, pp. 1-9,1994-07.

[5] K. Keeni, H. Shimodaira, T. Nishino, Y. Tan " Recognition of Devanagari Characters Using Neural Networks," *Transaction of Information and Systems Society of Japan*, IEICE TRANS.INF. &SYST., VOL. E79-D, No. 5,May 1996.

[6] K. Keeni, H. Shimodaira, K. Nakayama: On Distributed Representation of Output Layer for Recognizing Japanese Kana Characters Using Neural Networks; in *proceedings of the 4'th International Conference on Document Analysis and Recognition,ICDAR'97*, pp 600-604, 1997.

[7] David H. Kil, Frances B. Shin: Pattern recognition and Prediction with applications to Signal Characterization, *AIP PRESS*, 1996, pp. 134-138.

[8] K. Hara and K. Nakayama: Training Data Selection Method for generalization by multilayer Neural Networks;*Transaction of Information and Systems Society of Japan, Fundamentals*, VOL.E81-A, No.3,pp. 374-381, March 1998